STACK SAMPLE CODE

#include <stdio.h>

int MAXSIZE = 8;

int stack[8];

int top = -1;

int isempty() {

if(top == -1)

return 1;

else

return 0;

}

int isfull() {

if(top == MAXSIZE)

return 1;

else

return 0;

}

int peek() {

return stack[top];

}

int pop() {

int data;

if(!isempty()) {

data = stack[top];

top = top - 1;

return data;

} else {

printf("Could not retrieve data, Stack is empty.\n");

}

}

int push(int data) {

if(!isfull()) {

top = top + 1;

stack[top] = data;

} else {

printf("Could not insert data, Stack is full.\n");

}

}

int main() {

// push items on to the stack

push(3);

push(5);

push(9);

push(1);

push(12);

push(15);

printf("Element at top of the stack: %d\n" ,peek());

printf("Elements: \n");

// print stack data

while(!isempty()) {

int data = pop();

printf("%d\n",data);

}

printf("Stack full: %s\n" , isfull()?"true":"false");

printf("Stack empty: %s\n" , isempty()?"true":"false");

return 0;

}

If we compile and run the above program, it will produce the following result −

AD

Output

Element at top of the stack: 15

Elements:

15

12

1

9

5

3

Stack full: false

Stack empty: true

2.

|  |
| --- |
| /\* C++ program to implement basic stack     operations \*/  #include <bits/stdc++.h>    using namespace std;    #define MAX 1000    class Stack {      int top;    public:      int a[MAX]; // Maximum size of Stack        Stack() { top = -1; }      bool push(int x);      int pop();      int peek();      bool isEmpty();  };    bool Stack::push(int x)  {      if (top >= (MAX - 1)) {          cout << "Stack Overflow";          return false;      }      else {          a[++top] = x;          cout << x << " pushed into stack\n";          return true;      }  }    int Stack::pop()  {      if (top < 0) {          cout << "Stack Underflow";          return 0;      }      else {          int x = a[top--];          return x;      }  }  int Stack::peek()  {      if (top < 0) {          cout << "Stack is Empty";          return 0;      }      else {          int x = a[top];          return x;      }  }    bool Stack::isEmpty()  {      return (top < 0);  }    // Driver program to test above functions  int main()  {      class Stack s;      s.push(10);      s.push(20);      s.push(30);      cout << s.pop() << " Popped from stack\n";        //print top element of stack after poping      cout << "Top element is : " << s.peek() << endl;        //print all elements in stack :      cout<<"Elements present in stack : ";      while(!s.isEmpty())      {          // print top element in stack          cout<<s.peek()<<" ";          // remove top element in stack          s.pop();      }        return 0;  } |

**Output**

10 pushed into stack

20 pushed into stack

30 pushed into stack

30 Popped from stack

Top element is : 20

Elements present in stack : 20 10

3.

|  |
| --- |
| // C++ program for linked list implementation of stack  #include <bits/stdc++.h>  using namespace std;    // A structure to represent a stack  class StackNode {  public:      int data;      StackNode\* next;  };    StackNode\* newNode(int data)  {      StackNode\* stackNode = new StackNode();      stackNode->data = data;      stackNode->next = NULL;      return stackNode;  }    int isEmpty(StackNode\* root)  {      return !root;  }    void push(StackNode\*\* root, int data)  {      StackNode\* stackNode = newNode(data);      stackNode->next = \*root;      \*root = stackNode;      cout << data << " pushed to stack\n";  }    int pop(StackNode\*\* root)  {      if (isEmpty(\*root))          return INT\_MIN;      StackNode\* temp = \*root;      \*root = (\*root)->next;      int popped = temp->data;      free(temp);        return popped;  }    int peek(StackNode\* root)  {      if (isEmpty(root))          return INT\_MIN;      return root->data;  }    // Driver code  int main()  {      StackNode\* root = NULL;        push(&root, 10);      push(&root, 20);      push(&root, 30);        cout << pop(&root) << " popped from stack\n";        cout << "Top element is " << peek(root) << endl;        cout<<"Elements present in stack : ";       //print all elements in stack :      while(!isEmpty(root))      {          // print top element in stack          cout<<peek(root)<<" ";          // remove top element in stack          pop(&root);      }        return 0;  }    // This is code is contributed by rathbhupendra |

**Output**

10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack

Top element is 20

Elements present in stack : 20 10